IBM DATA SCIENCE CAPSTONE PROJECT

SPACE X FALCON 9 LANDING ANALYSIS

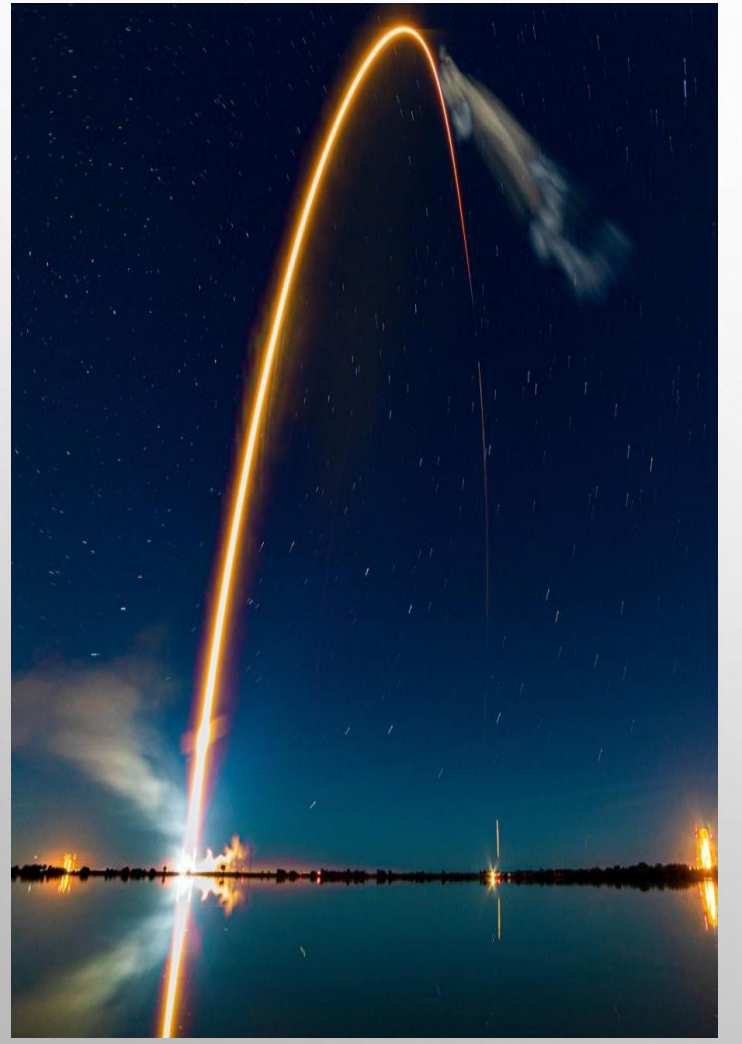
BHAVEEN MORAR

03/05/2022

OUTLINE

- EXECUTIVE SUMMARY
- INTRODUCTION
- METHODOLOGY
- RESULTS
  - VISUALIZATION – CHARTS
  - DASHBOARD
- DISCUSSION
  - FINDINGS & IMPLICATIONS
- CONCLUSION
- APPENDIX

EXECUTIVE SUMMARY



SUMMARY OF METHODOLOGIES:

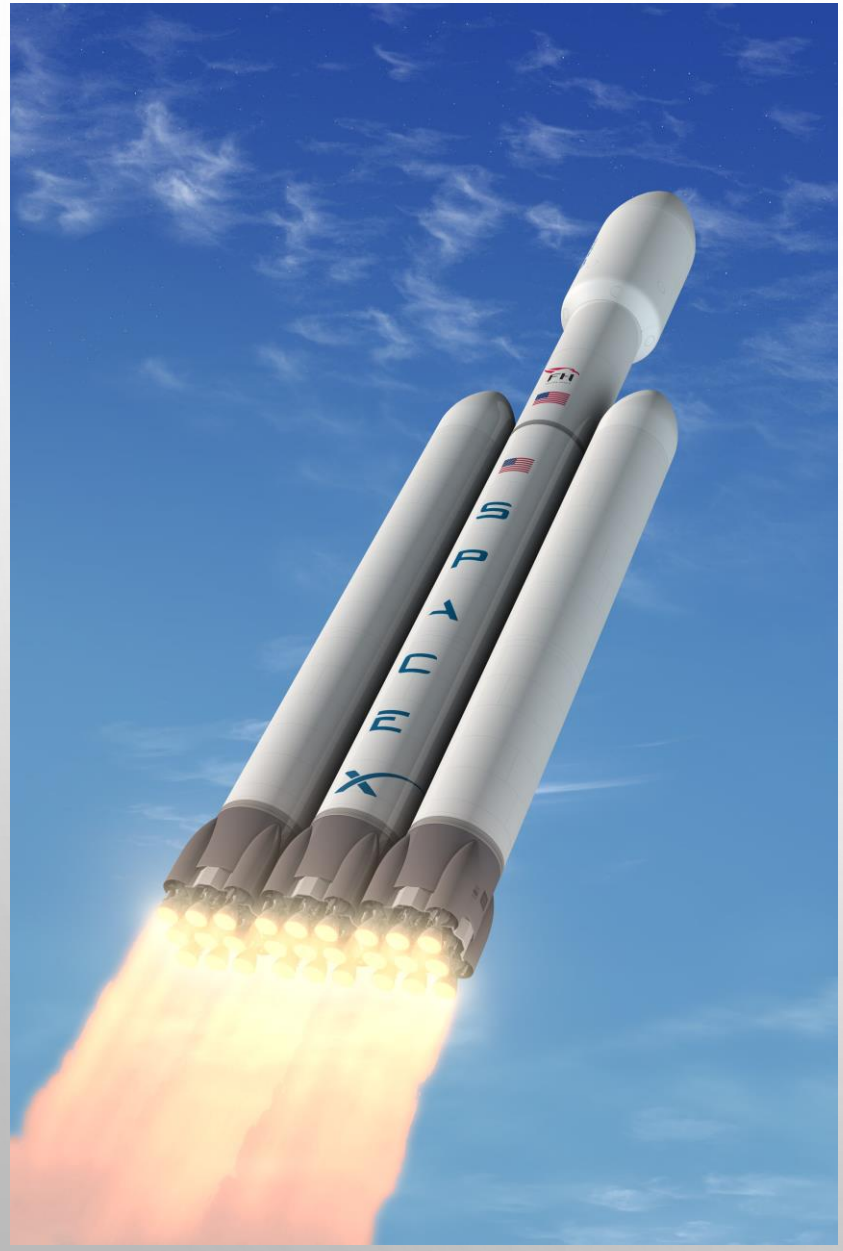
This project follows these steps:

- Data collection
- Data wrangling
- Exploratory data analysis
- Data visualization
- Machine learning prediction

SUMMARY OF RESULTS:

This project produced the following outputs and visualizations:

- Exploratory data analysis (EDA) results
- Geospatial analytics
- Interactive dashboard
- Predictive analysis of classification models



INTRODUCTION

- SpaceX launches Falcon 9 rockets at a cost of around $62m. This is considerably cheaper than other providers (which usually cost upwards of $165m), and much of the savings are because SpaceX can land, and then re-use the first stage of the rocket.
- If we can make predictions on whether the first stage will land, we can determine the cost of a launch, and use this information to assess whether or not an alternate company should bid and SpaceX for a rocket launch.
- This project will ultimately predict if the Space X Falcon 9 first stage will land successfully.

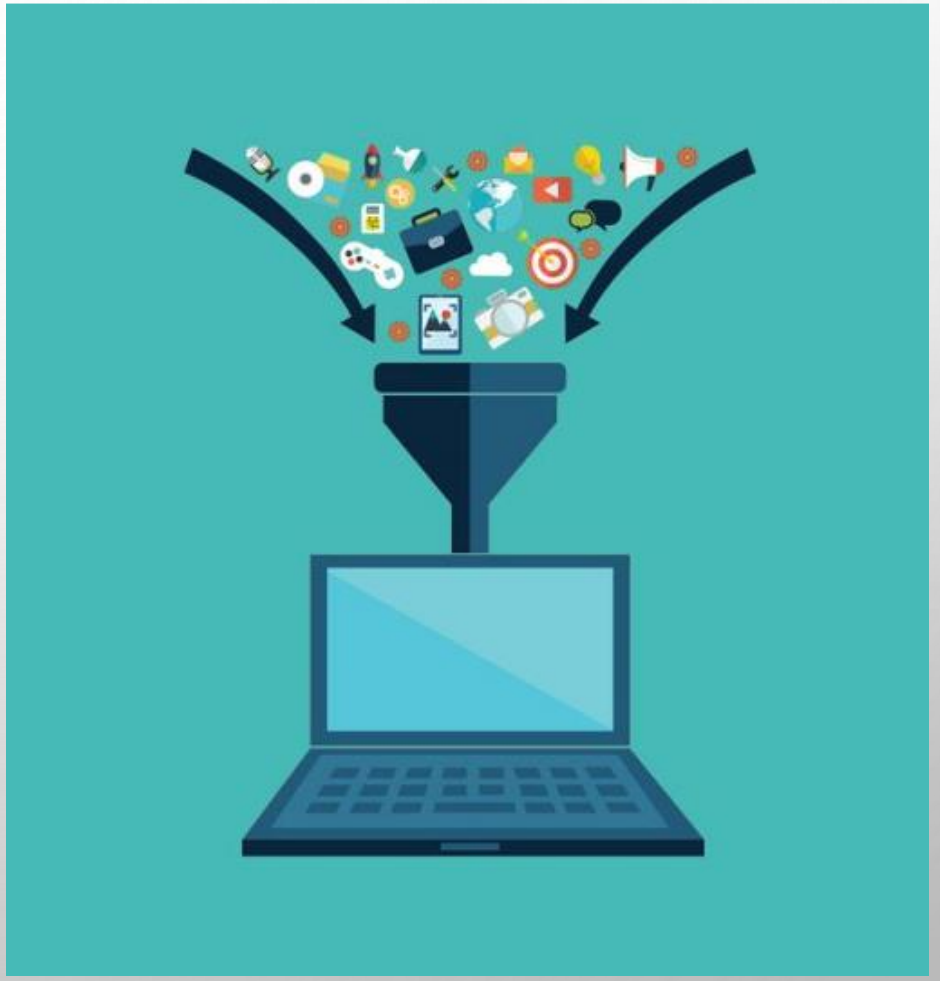
METHODOLOGY

1. **DATA COLLECTION**
   - Making GET requests to the spacex REST API
   - Web scraping
2. **DATA WRANGLING**
   - Removed nan values using the .Fillna() method to
   - Used the method value_counts() to determine :
     - The number of launches on each site
     - The number and occurrence of each orbit
     - The number and occurrence of mission outcome per orbit type
   - Created a new landing outcome column that shows:
     - The booster landed successfully: '1'
     - The booster did not land successfully: '0'
3. **EXPLORATORY DATA ANALYSIS**
   - Used SQL queries to evaluate the spacex dataset
   - We visualized relationships between variables and determine patterns using matplotlib.
4. **INTERACTIVE VISUAL ANALYTICS**
   - Geospatial analytics using folium
   - Creating an interactive dashboard using plotly dash
5. **DATA MODELLING AND EVALUATION**
   - Using scikit-learn to:
     - Pre-process (standardize) the data
     - Split the data into training and testing data using train/test_split
     - Train different classification models
     - Find hyperparameters using gridsearchcv
   - Plotting confusion matrices for each classification model
   - Assessing the accuracy of each classification model

DATA COLLECTION

The SpaceX API was used to retrieve data about launches. This includes information about the rocket used, payload delivered, launch specifications, landing specifications, and landing outcome.

- Make a GET response to the SpaceX REST API
  Convert the response to a .json file then to a Pandas DataFrame
- Use custom logic to clean the data
  Define lists for data to be stored in
  Call custom functions to retrieve data and fill the lists
  Use these lists as values in a dictionary and construct the dataset
- Create a Pandas DataFrame from the constructed dictionary dataset
- Filter the DataFrame to only include Falcon 9 launches
  Reset the FlightNumber column
  Replace missing values of PayloadMass with the mean PayloadMass value
- Link to workbook here



DATA COLLECTION

Web scraping was used to collect Falcon 9 historical launch records from a Wikipedia page titled List of Falcon 9 and Falcon Heavy launches.

- Request the HTML page from the static URL
  Assign the response to an object
- Create a BeautifulSoup object from the HTML response object
  Find all tables within the HTML page
- Collect all column header names from the tables found within the HTML page
- Use the column names as keys in a dictionary
  Use custom functions and logic to parse all launch tables to fill the dictionary values
- Convert the dictionary to a Pandas DataFrame ready for export
- Link to workbook here

DATA WRANGLING

```
# Apply value_counts() on column LaunchSite
df['LaunchSite'].value_counts()
```

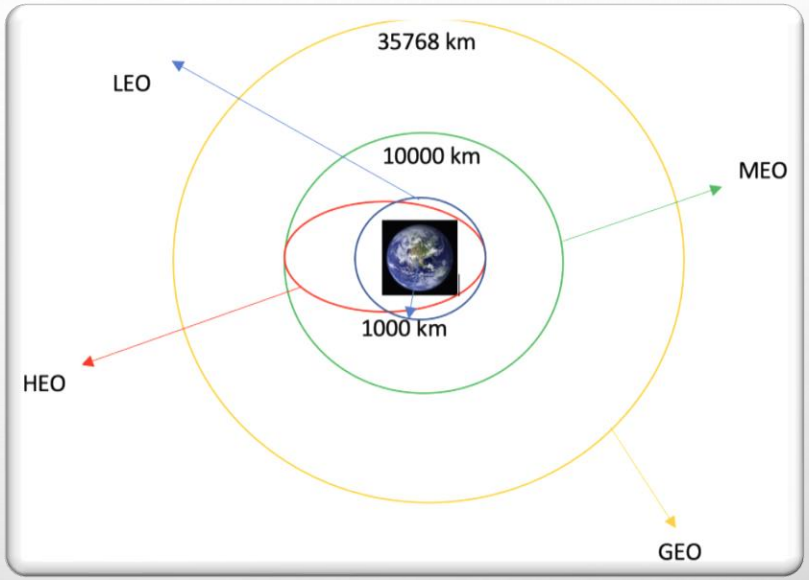
```
CCAFS SLC 40    55
KSC LC 39A      22
VAFB SLC 4E     13
Name: LaunchSite, dtype: int64
```

`LaunchSite` column

- Each launch aims to a dedicated orbit, and some of the common orbit types are shown in the figure below. The orbit type is in the `Orbit` column.

```
# Apply value_counts on Orbit column
df['Orbit'].value_counts()
```

```
GTO     27
ISS     21
VLEO    14
PO       9
LEO      7
SSO      5
MEO      3
ES-L1    1
HEO      1
SO       1
GEO      1
Name: Orbit, dtype: int64
```




```
# landing_outcomes = values on Outcome column
landing_outcomes = df['Outcome'].value_counts()
landing_outcomes
```

```
True ASDS      41
None None      19
True RTLS      14
False ASDS      6
True Ocean      5
False Ocean     2
None ASDS       2
False RTLS      1
Name: Outcome, dtype: int64
```

Initial Data Exploration:

- Using the `.value_counts()` method to determine the following:
  1. Number of launches on each site
  2. Number and occurrence of each orbit
  3. Number and occurrence of landing outcome per orbit type

DATA WRANGLING – CONTINUED

Context:

- The landing outcome is shown in the `Outcome` column:

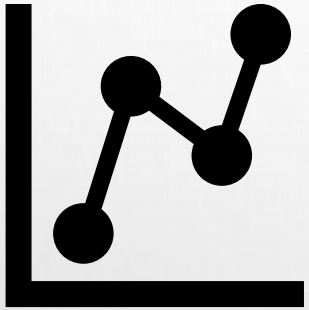
`True Ocean` – the mission outcome was successfully landed to a specific region of the ocean
`False Ocean` – the mission outcome was unsuccessfully landed to a specific region of the ocean.
`True RTLS` – the mission outcome was successfully landed to a ground pad
`False RTLS` – the mission outcome was unsuccessfully landed to a ground pad.
`True ASDS` – the mission outcome was successfully landed to a drone ship
`False ASDS` – the mission outcome was unsuccessfully landed to a drone ship.
`None ASDS` and `None None` – these represent a failure to land.

Data Wrangling:

- To determine whether a booster will successfully land, it is best to have a binary column, i.e., where the value is 1 or 0, representing the success of the landing.
- This is done by:
  1. Defining a set of unsuccessful (bad) outcomes, `bad_outcome`
  2. Creating a list, `landing_class`, where the element is 0 if the corresponding row in `Outcome` is in the set `bad_outcome`, otherwise, it's 1.
  3. Create a `Class` column that contains the values from the list `landing_class`
  4. Export the DataFrame as a .csv file.

EXPLORATORY DATA ANALYSIS

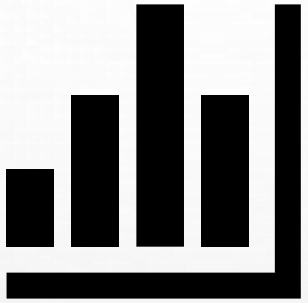
SCATTER CHARTS



Scatter charts were produced to visualize the relationships between:

- Flight Number and Launch Site
- Payload and Launch Site
- Orbit Type and Flight Number
- Payload and Orbit Type

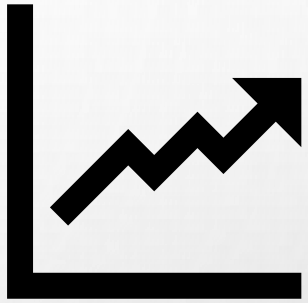
BAR CHART



A bar chart was produced to visualize the relationship between:

Success Rate and Orbit Type

LINE CHARTS

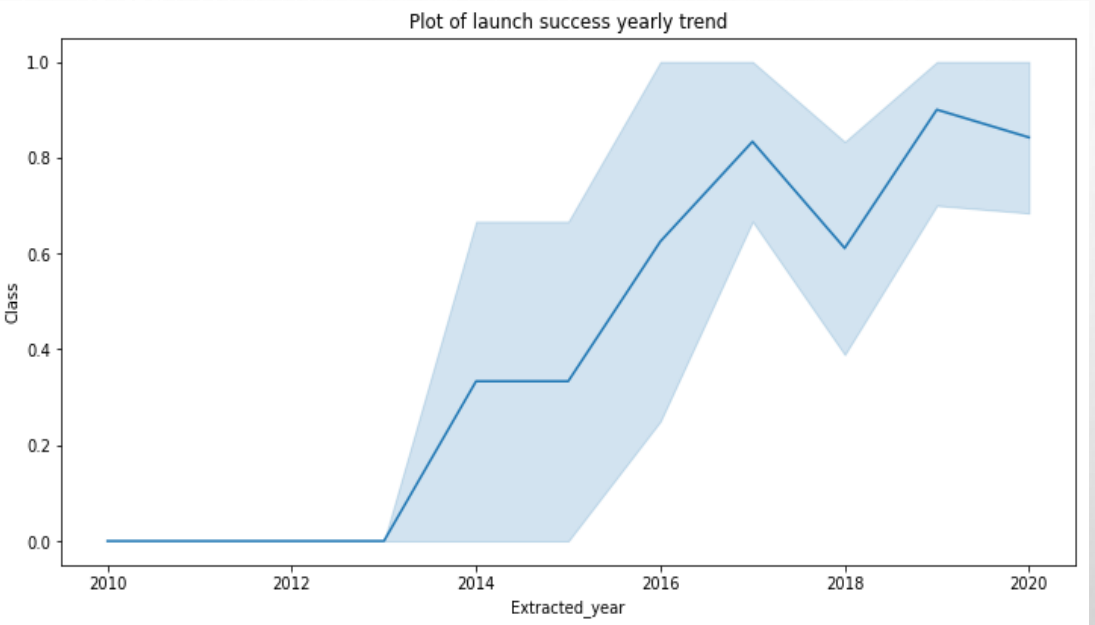


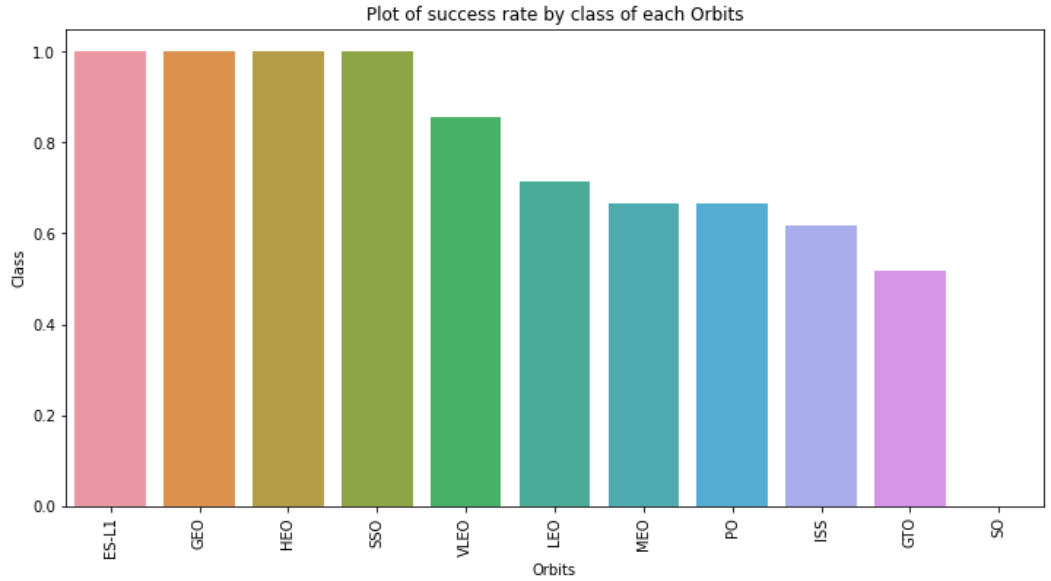
Line charts were produced to visualize the relationships between:

Success Rate and Year (i.e. the launch success yearly trend)

EDA WITH DATA VISUALIZATION

- We explored the data by visualizing the relationship between flight number and launch site, payload and launch site, success rate of each orbit type, flight number and orbit type, the launch success yearly trend.





- The link to the notebook can be found here

DATA VISUALIZATION WITH SQL

To gather some information about the dataset, some SQL queries were performed.

The SQL queries performed on the data set were used to:

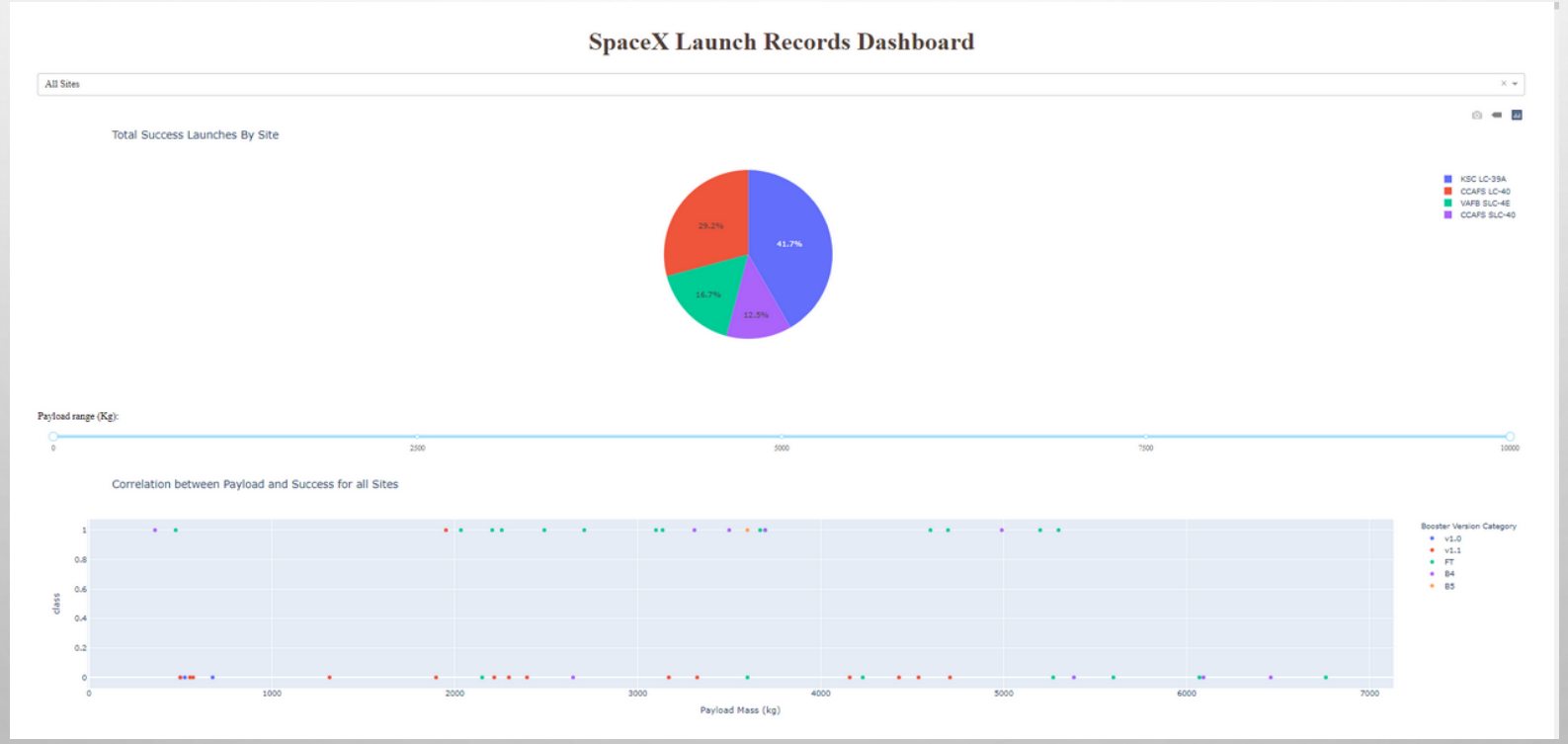
1. Display the names of the unique launch sites in the space mission
2. Display 5 records where launch sites begin with the string 'CCA'
3. Display the total payload mass carried by boosters launched by NASA (CRS)
4. Display the average payload mass carried by booster version F9 v1.1
5. List the date when the first successful landing outcome on a ground pad was achieved
6. List the names of the boosters which had success on a drone ship and a payload mass between 4000 and 6000 kg
7. List the total number of successful and failed mission outcomes
8. List the names of the booster versions which have carried the maximum payload mass
9. List the failed landing outcomes on drone ships, their booster versions, and launch site names for 2015
10. Rank the count of landing outcomes (such as Failure (drone ship) or Success (ground pad)) between the date 2010-06-04 and 2017-03-20, in descending order

GEOSPACIAL ANALYSIS WITH FOLIUM

- We marked all launch sites, and added map objects such as markers, circles, lines to mark the success or failure of launches for each site on the folium map.
- We assigned the feature launch outcomes (failure or success) to class 0 and 1.I.E., 0 for failure, and 1 for success.
- Using the color-labeled marker clusters, we identified which launch sites have relatively high success rate.
- We calculated the distances between a launch site to its proximities. We answered some question for instance:
  - Are launch sites near railways, highways and coastlines.
  - Do launch sites keep certain distance away from cities.

DASHBOARD APPLICATION WITH PLOTLY DASH

- We built an interactive dashboard with plotly dash
- We plotted pie charts showing the total launches by a certain sites
- We plotted scatter graph showing the relationship with outcome and payload mass (kg) for the different booster version.



PREDICTIVE ANALYSIS - CLASSIFICATION

- The following steps were taking to develop, evaluate, and find the best performing classification model:

Model Development

- To prepare the dataset for model development:
  - Load dataset
  - Perform necessary data transformations (standardise and pre-process)
  - Split data into training and test data sets, using `train_test_split()`
  - Decide which type of machine learning algorithms are most appropriate
- For each chosen algorithm:
  - Create a `GridSearchCV` object and a dictionary of parameters
  - Fit the object to the parameters
  - Use the training data set to train the model

Model Evaluation

- For each chosen algorithm:
  - Using the output GridSearchCV object:
    - Check the tuned hyperparameters (`best_params_`)
    - Check the accuracy (`score` and `best_score_`)
  - Plot and examine the Confusion Matrix

Selecting Best Classification Model

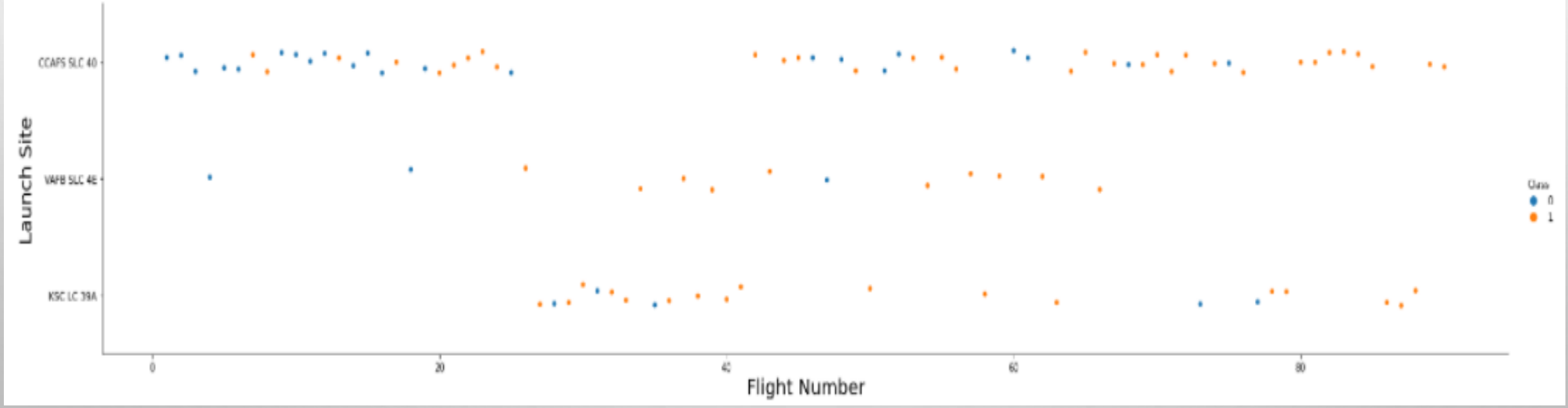
- Review the accuracy scores for all chosen algorithms
- The model with the highest accuracy score is determined as the best performing model

RESULTS

- Exploratory data analysis results
- Interactive analytics demo in screenshots
- Predictive analysis results

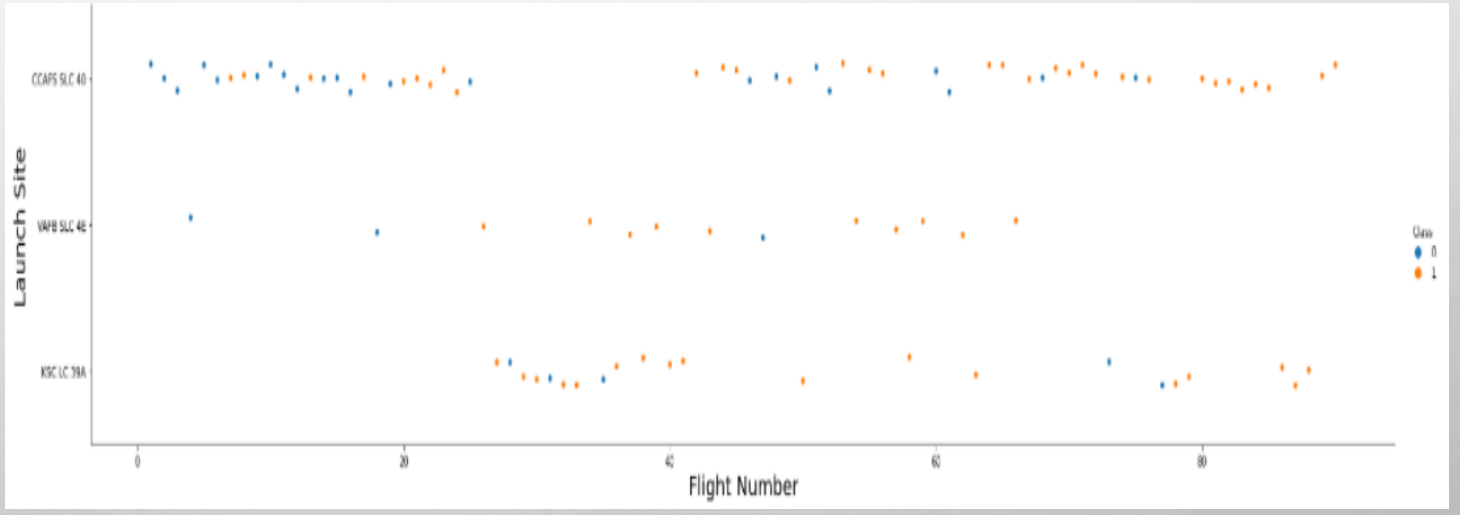
FLIGHT NUMBER VS. LAUNCH SITE

From the plot, we found that the larger the flight amount at a launch site, the greater the success rate at a launch site.

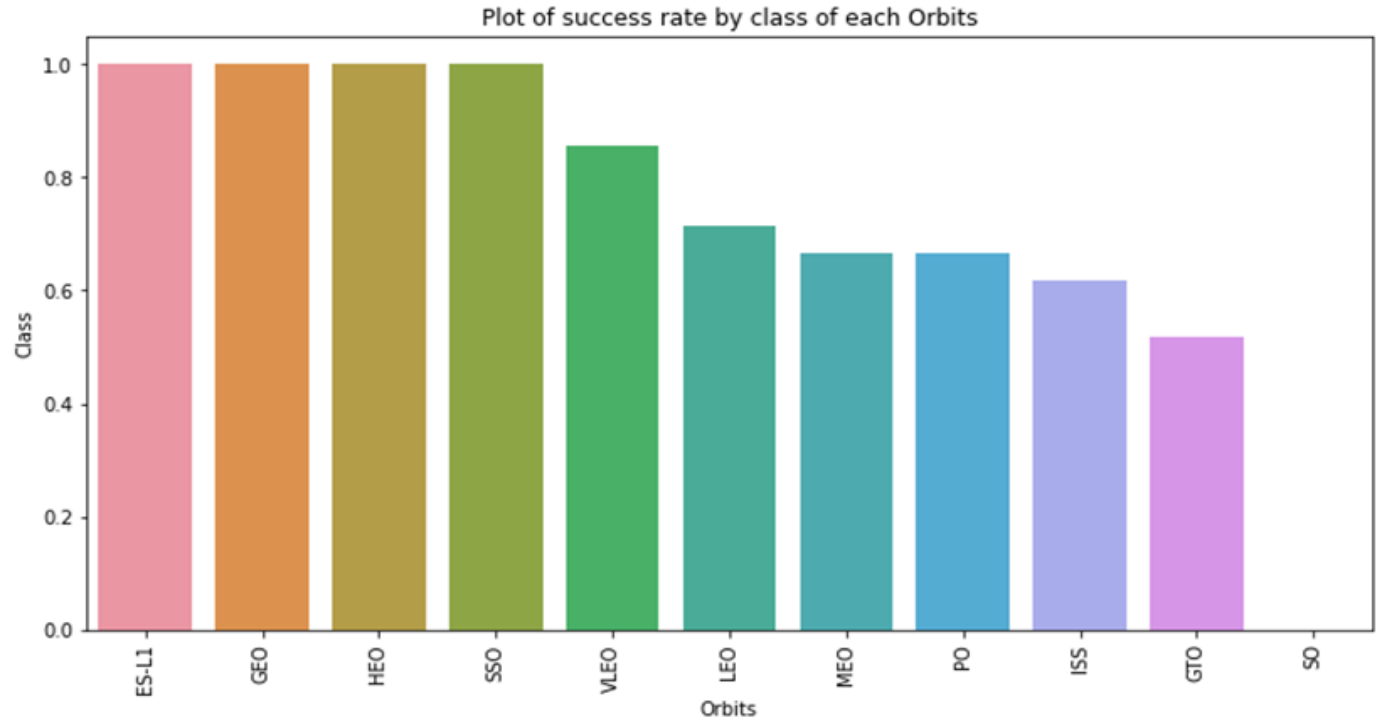


Payload vs. Launch Site

The greater the payload mass for launch site CCAFS SLC 40 the higher the success rate for the rocket.



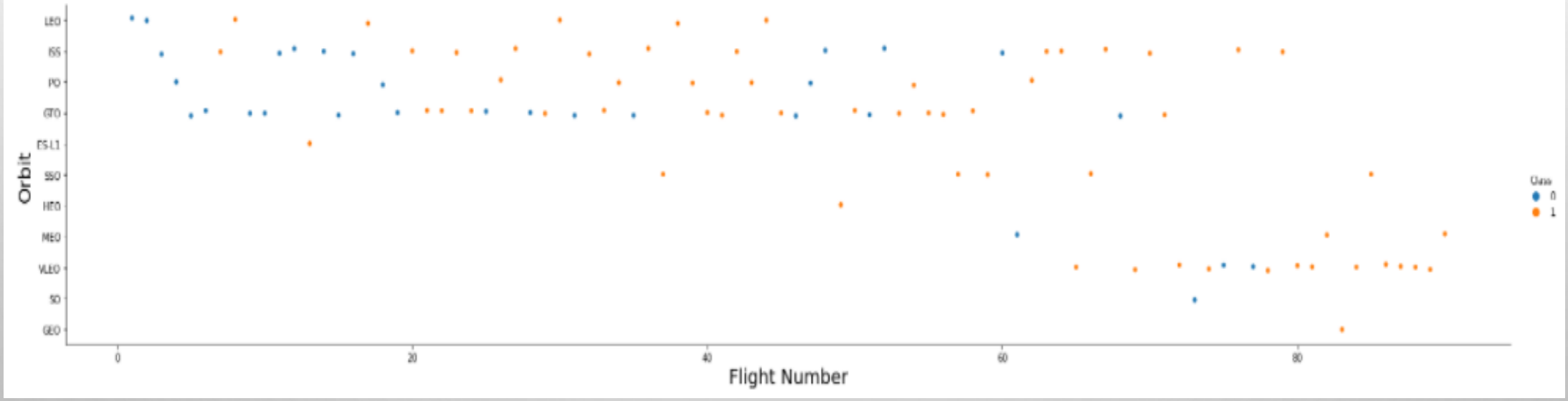
Success Rate vs. Orbit Type



From the plot, we can see that ES-L1, GEO, HEO, SSO, VLEO had the most success rate.

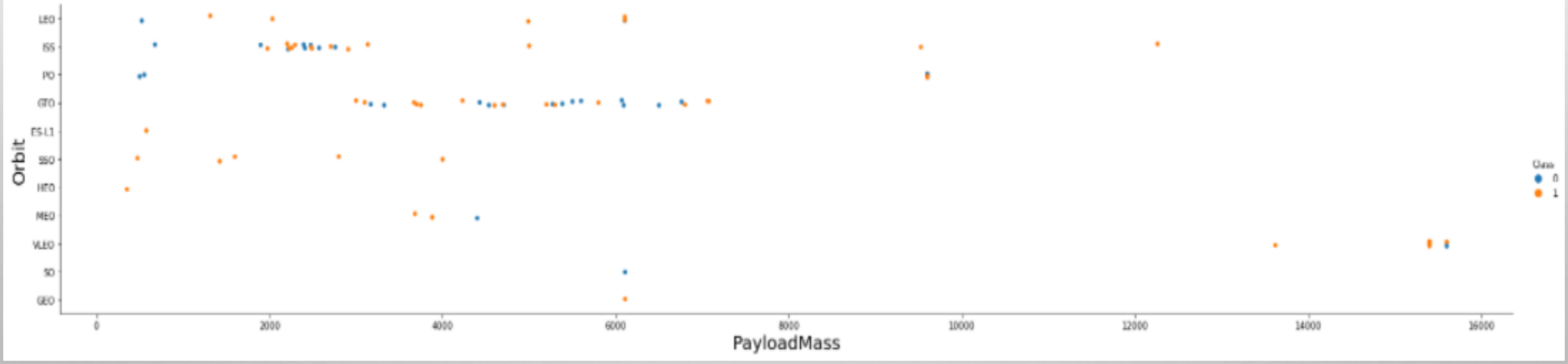
Flight Number vs. Orbit Type

The plot below shows the Flight Number vs. Orbit type. We observe that in the LEO orbit, success is related to the number of flights whereas in the GTO orbit, there is no relationship between flight number and the orbit.



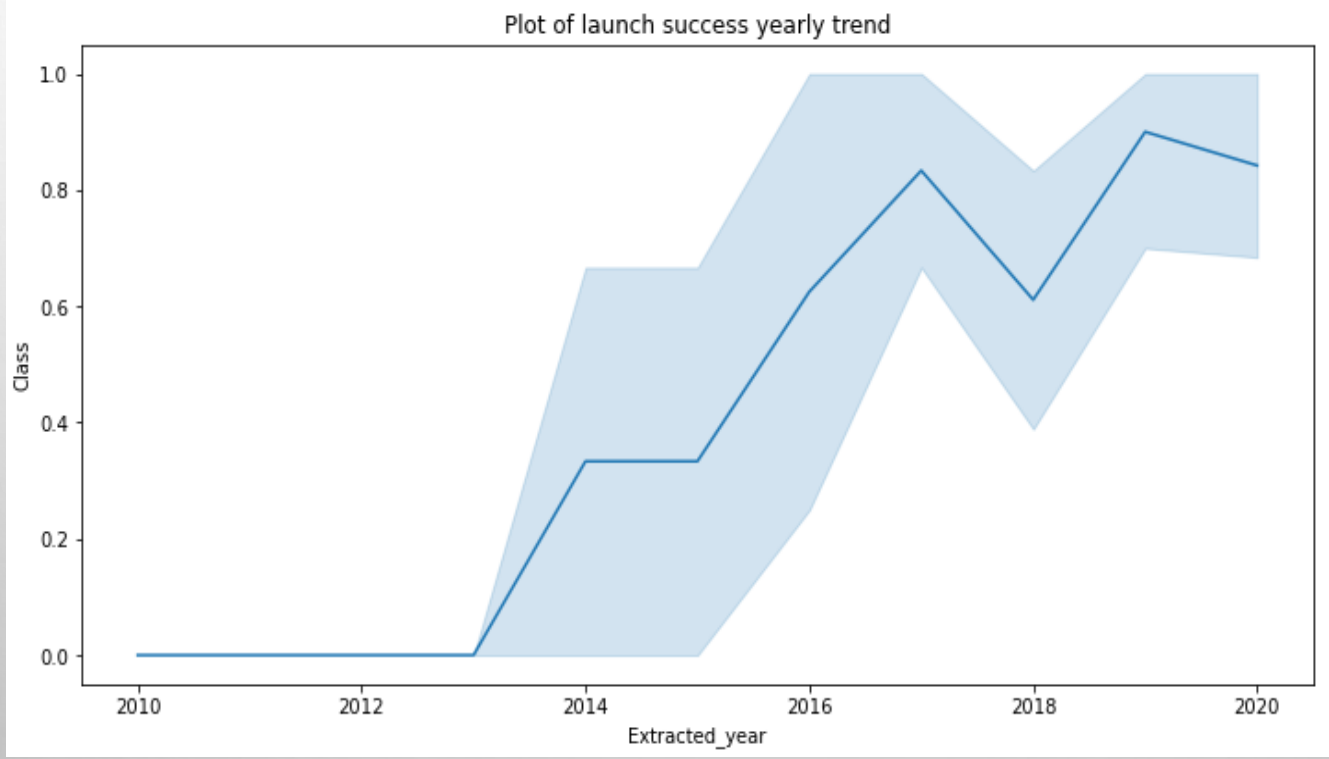
Payload vs. Orbit Type

We can observe that with heavy payloads, the successful landing are more for PO, LEO and ISS orbits.



Launch Success Yearly Trend

From the plot, we can observe that success rate since 2013 kept on increasing till 2020.



All Launch Site Names

We used the key word **DISTINCT** to show only unique launch sites from the SpaceX data.

.

Display the names of the unique launch sites in the space mission

In [10]:
```
task_1 = '''
        SELECT DISTINCT LaunchSite
        FROM SpaceX
'''
create_pandas_df(task_1, database=conn)
```

Out[10]:

	launchsite
0	KSC LC-39A
1	CCAFS LC-40
2	CCAFS SLC-40
3	VAFB SLC-4E

Launch Site Names Begin with 'CCA'

Display 5 records where launch sites begin with the string 'CCA'

In [11]:
```
task_2 = '''
        SELECT *
        FROM SpaceX
        WHERE LaunchSite LIKE 'CCA%'
        LIMIT 5
        '''
create_pandas_df(task_2, database=conn)
```

Out[11]:

	date	time	boosterversion	launchsite	payload	payloadmasskg	orbit	customer	missionoutcome	landingoutcome
0	2010-04-06	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
1	2010-08-12	15:43:00	F9 v1.0 B0004	CCAFS LC-40	Dragon demo flight C1, two CubeSats, barrel of...	0	LEO (ISS)	NASA (COTS) NRO	Success	Failure (parachute)
2	2012-05-22	07:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
3	2012-08-10	00:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
4	2013-01-03	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No attempt

We used the query above to display 5 records where launch sites begin with `CCA`

.

Total Payload Mass

We calculated the total payload carried by boosters from NASA as 45596 using the query below

Display the total payload mass carried by boosters launched by NASA (CRS)

In [12]:
```
task_3 = '''
        SELECT SUM(PayloadMassKG) AS Total_PayloadMass
        FROM SpaceX
        WHERE Customer LIKE 'NASA (CRS)'
        '''
create_pandas_df(task_3, database=conn)
```

Out[12]:

	total_payloadmass
0	45596

Average Payload Mass by F9 v1.1

We calculated the average payload mass carried by booster version F9 v1.1 as 2928.4

Display average payload mass carried by booster version F9 v1.1

In [13]:
```
task_4 = '''
        SELECT AVG(PayloadMassKG) AS Avg_PayloadMass
        FROM SpaceX
        WHERE BoosterVersion = 'F9 v1.1'
        '''
create_pandas_df(task_4, database=conn)
```

Out[13]:

	avg_payloadmass
0	2928.4

First Successful Ground Landing Date

We observed that the dates of the first successful landing outcome on ground pad was 22nd December 2015

.

In [14]:
```
task_5 = '''
        SELECT MIN(Date) AS FirstSuccessfull_landing_date
        FROM SpaceX
        WHERE LandingOutcome LIKE 'Success (ground pad)'
        '''
create_pandas_df(task_5, database=conn)
```

Out[14]:

	firstsuccessfull_landing_date
0	2015-12-22

Successful Drone Ship Landing with Payload between 4000 and 6000

In [15]:

```
task_6 = '''
        SELECT BoosterVersion
        FROM SpaceX
        WHERE LandingOutcome = 'Success (drone ship)'
            AND PayloadMassKG > 4000
            AND PayloadMassKG < 6000
        '''
create_pandas_df(task_6, database=conn)
```

Out[15]:

	boosterversion
0	F9 FT B1022
1	F9 FT B1026
2	F9 FT B1021.2
3	F9 FT B1031.2

We used the WHERE clause to filter for boosters which have successfully landed on drone ship and applied the AND condition to determine successful landing with payload mass greater than 4000 but less than 6000

Total Number of Successful and Failure Mission Outcomes

List the total number of successful and failure mission outcomes

In [16]:
```
task_7a = '''
        SELECT COUNT(MissionOutcome) AS SuccessOutcome
        FROM SpaceX
        WHERE MissionOutcome LIKE 'Success%'
        '''

task_7b = '''
        SELECT COUNT(MissionOutcome) AS FailureOutcome
        FROM SpaceX
        WHERE MissionOutcome LIKE 'Failure%'
        '''
print('The total number of successful mission outcome is:')
display(create_pandas_df(task_7a, database=conn))
print()
print('The total number of failed mission outcome is:')
create_pandas_df(task_7b, database=conn)
```

Out[16]:

The total number of successful mission outcome is:

	successoutcome
0	100

The total number of failed mission outcome is:

	failureoutcome
0	1

We used wildcard like '%' to filter for WHERE MissionOutcome was a success or a failure.

Boosters Carried Maximum Payload

We determined the booster that have carried the maximum payload using a subquery in the **WHERE** clause and the **MAX()** function.

List the names of the booster_versions which have carried the maximum payload mass. Use a subquery

In [17]:

```
task_8 = '''
        SELECT BoosterVersion, PayloadMassKG
        FROM SpaceX
        WHERE PayloadMassKG = (
                                SELECT MAX(PayloadMassKG)
                                FROM SpaceX
                                )
        ORDER BY BoosterVersion
        '''
create_pandas_df(task_8, database=conn)
```

Out[17]:

	boosterversion	payloadmasskg
0	F9 B5 B1048.4	15600
1	F9 B5 B1048.5	15600
2	F9 B5 B1049.4	15600
3	F9 B5 B1049.5	15600
4	F9 B5 B1049.7	15600
5	F9 B5 B1051.3	15600
6	F9 B5 B1051.4	15600
7	F9 B5 B1051.6	15600
8	F9 B5 B1056.4	15600
9	F9 B5 B1058.3	15600
10	F9 B5 B1060.2	15600
11	F9 B5 B1060.3	15600

2015 Launch Records

We used a combinations of the WHERE clause, LIKE, AND, and BETWEEN conditions to filter for failed landing outcomes in drone ship, their booster versions, and launch site names for year 2015

List the failed landing_outcomes in drone ship, their booster versions, and launch site names for in year 2015

In [18]:
```
task_9 = '''
        SELECT BoosterVersion, LaunchSite, LandingOutcome
        FROM SpaceX
        WHERE LandingOutcome LIKE 'Failure (drone ship)'
            AND Date BETWEEN '2015-01-01' AND '2015-12-31'
        '''
create_pandas_df(task_9, database=conn)
```

Out[18]:

	boosterversion	launchsite	landingoutcome
0	F9 v1.1 B1012	CCAFS LC-40	Failure (drone ship)
1	F9 v1.1 B1015	CCAFS LC-40	Failure (drone ship)

Rank Landing Outcomes Between 2010-06-04 and 2017-03-20

We selected Landing outcomes and the COUNT of landing outcomes from the data and used the WHERE clause to filter for landing outcomes BETWEEN 2010-06-04 to 2010-03-20.

We applied the GROUP BY clause to group the landing outcomes and the ORDER BY clause to order the grouped landing outcome in descending order.

Rank the count of landing outcomes (such as Failure (drone ship) or Success (ground pad))

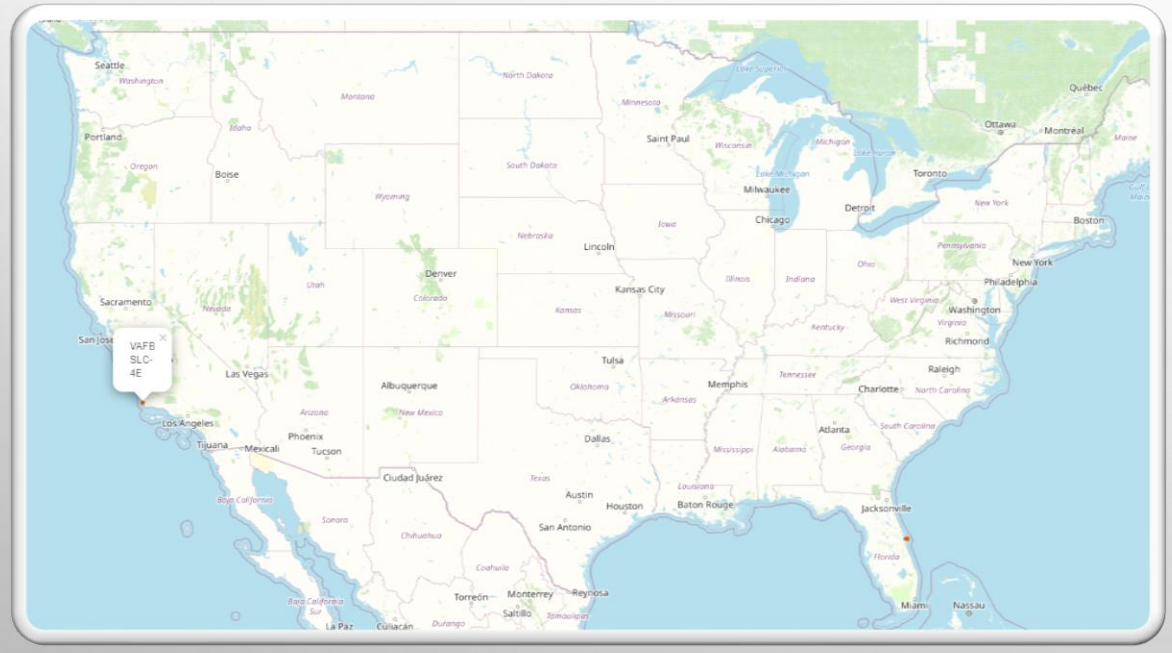
In [19]:
```
task_10 = '''
        SELECT LandingOutcome, COUNT(LandingOutcome)
        FROM SpaceX
        WHERE DATE BETWEEN '2010-06-04' AND '2017-03-20'
        GROUP BY LandingOutcome
        ORDER BY COUNT(LandingOutcome) DESC
        '''
create_pandas_df(task_10, database=conn)
```

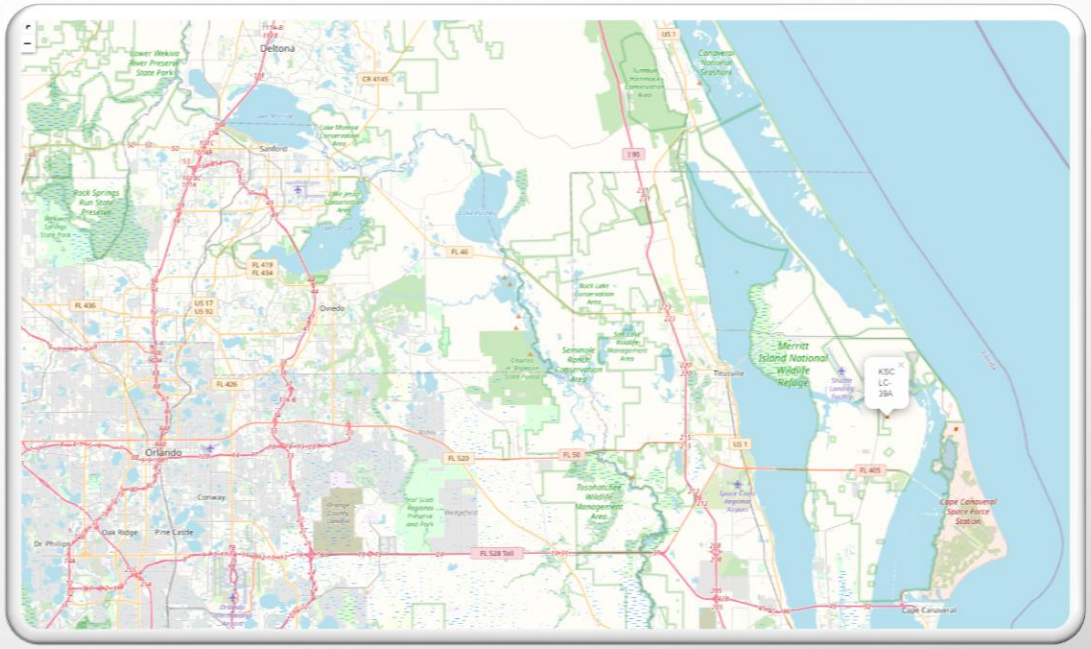
Out[19]:

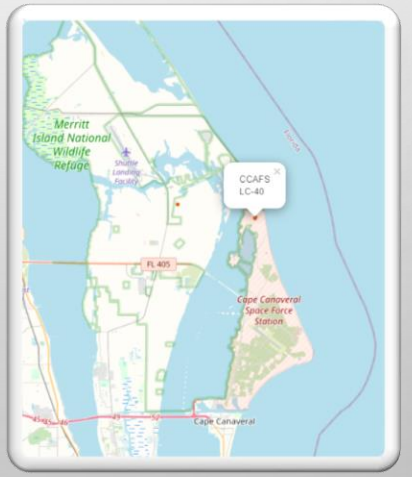
	landingoutcome	count
0	No attempt	10
1	Success (drone ship)	6
2	Failure (drone ship)	5
3	Success (ground pad)	5
4	Controlled (ocean)	3
5	Uncontrolled (ocean)	2
6	Precluded (drone ship)	1
7	Failure (parachute)	1

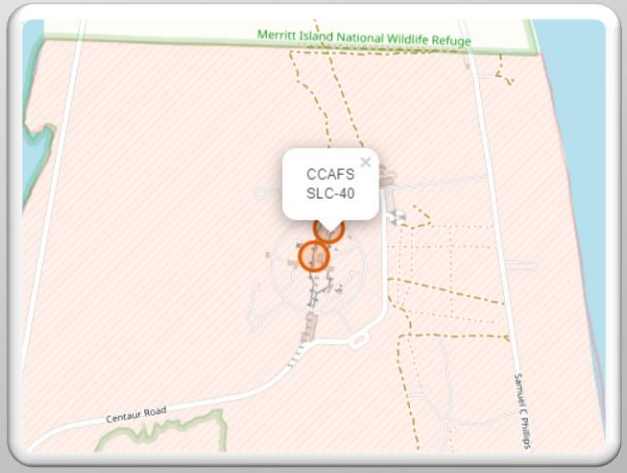
ALL LAUNCH SITES ON A MAP

All SpaceX launch sites are on coasts of the United States of America, specifically Florida and California.



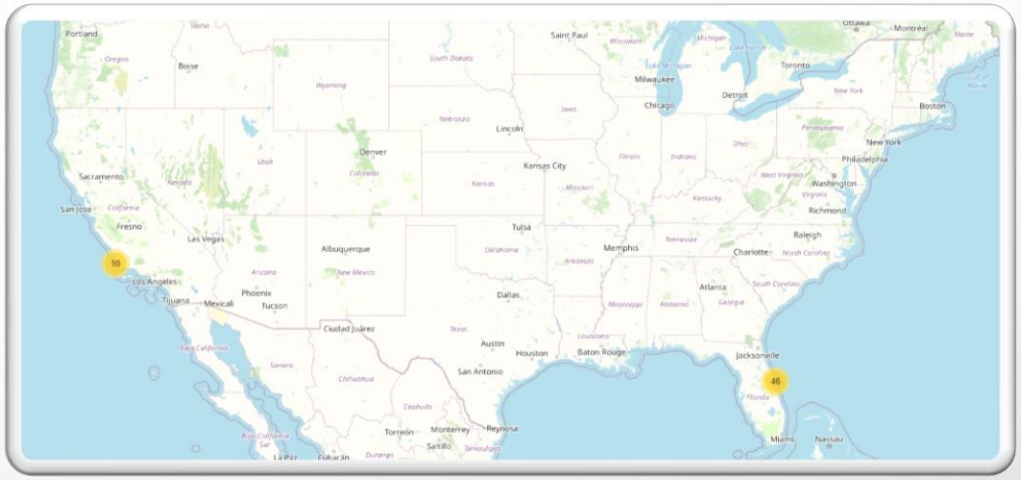






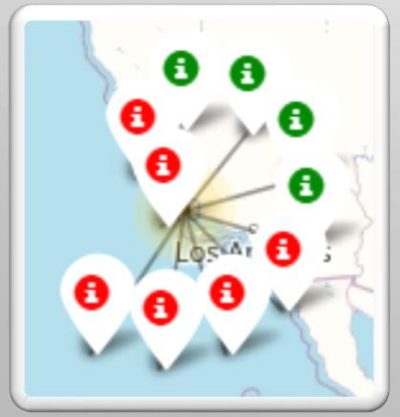
SUCCESS/FAILED LAUNCHES FOR EACH SITE

Launches have been grouped into clusters, and annotated with green icons for successful launches, and red icons for failed launches.

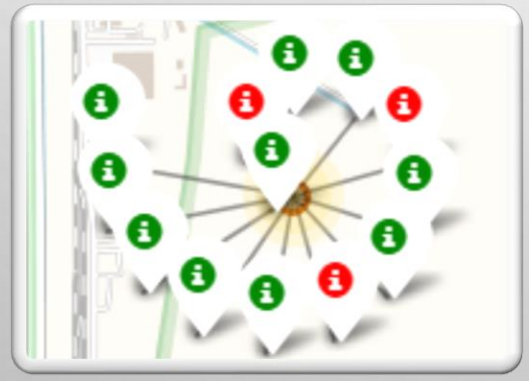


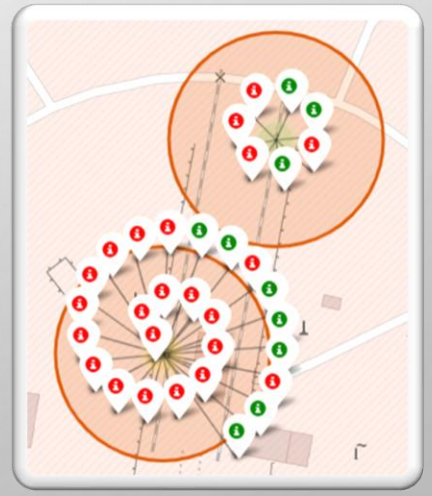
CCAFS SLC-40 and CCAFS LC-40

VAFB SLC-4E

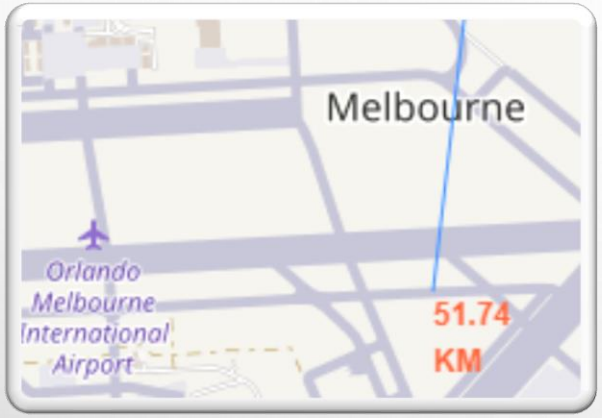


KSC LC-39A





PROXIMITY OF LAUNCH SITES TO OTHER POINTS OF INTEREST



Using the CCAFS SLC-40 launch site as an example site, we can understand more about the placement of launch sites.

Are launch sites in close proximity to railways?

- YES. The coastline is only 0.87 km due East.

Are launch sites in close proximity to highways?

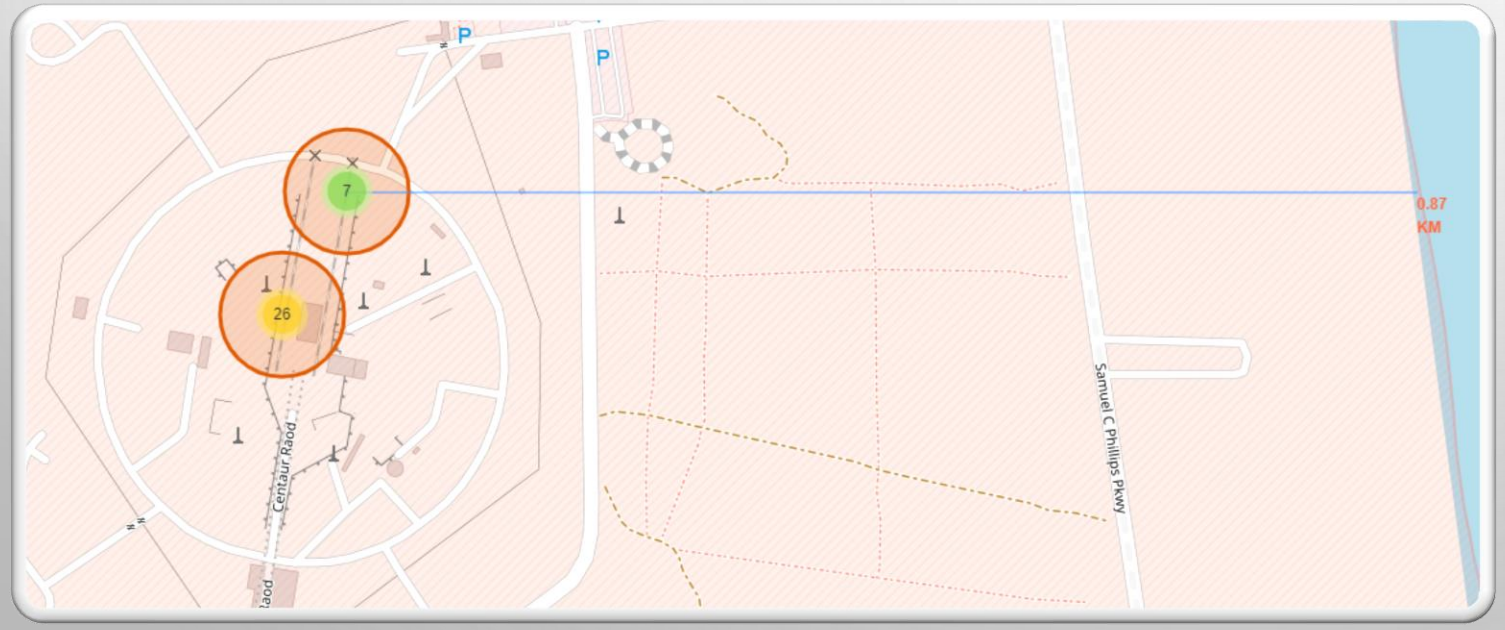
- YES. The nearest highway is only 0.59km away.

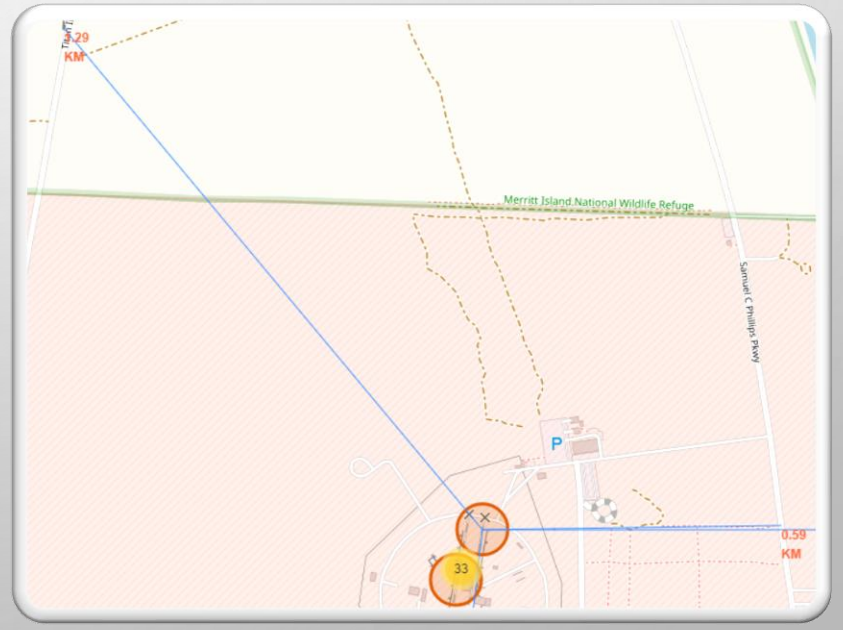
Are launch sites in close proximity to railways?

- YES. The nearest railway is only 1.29 km away.

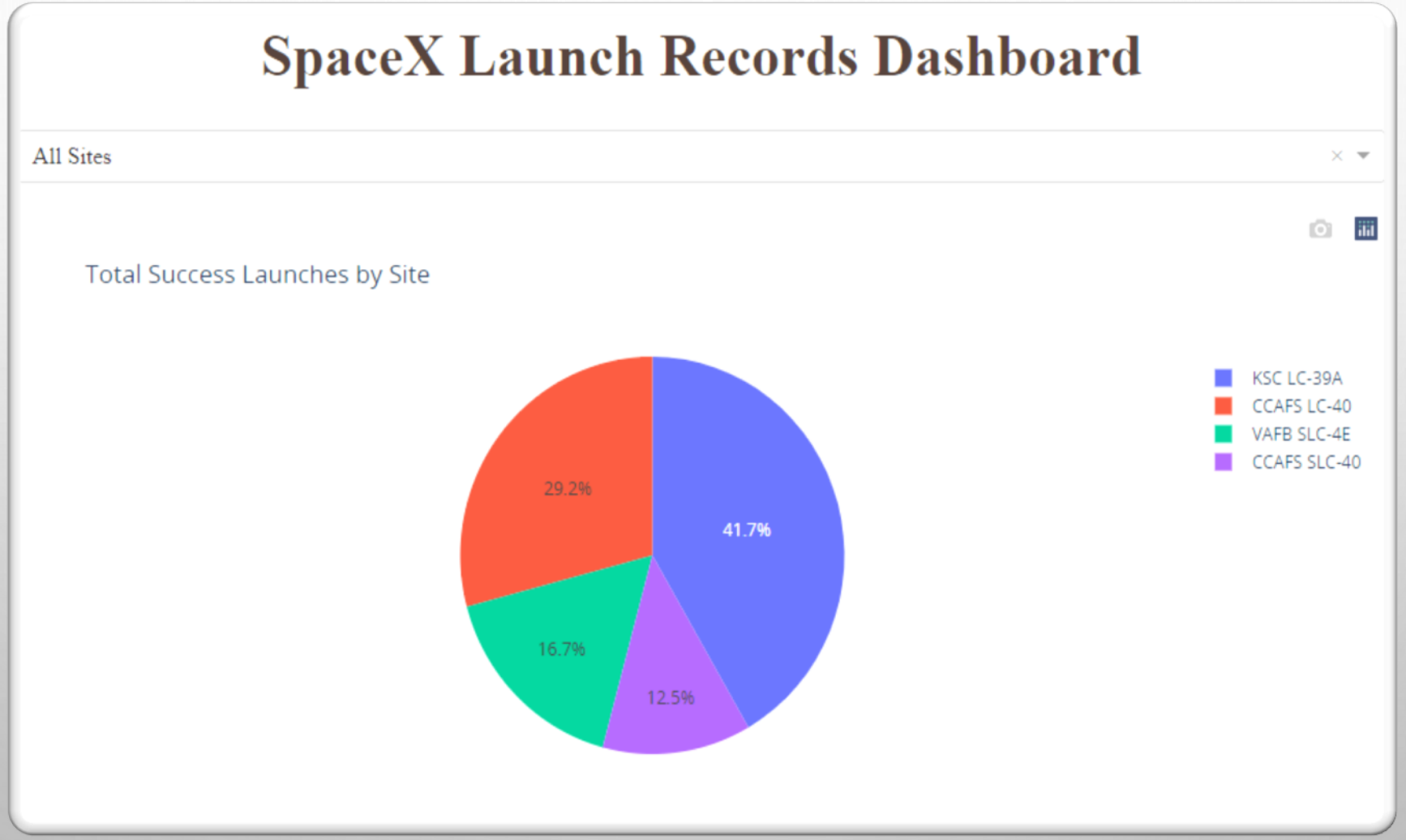
Do launch sites keep certain distance away from cities?

- YES. The nearest city is 51.74 km away.

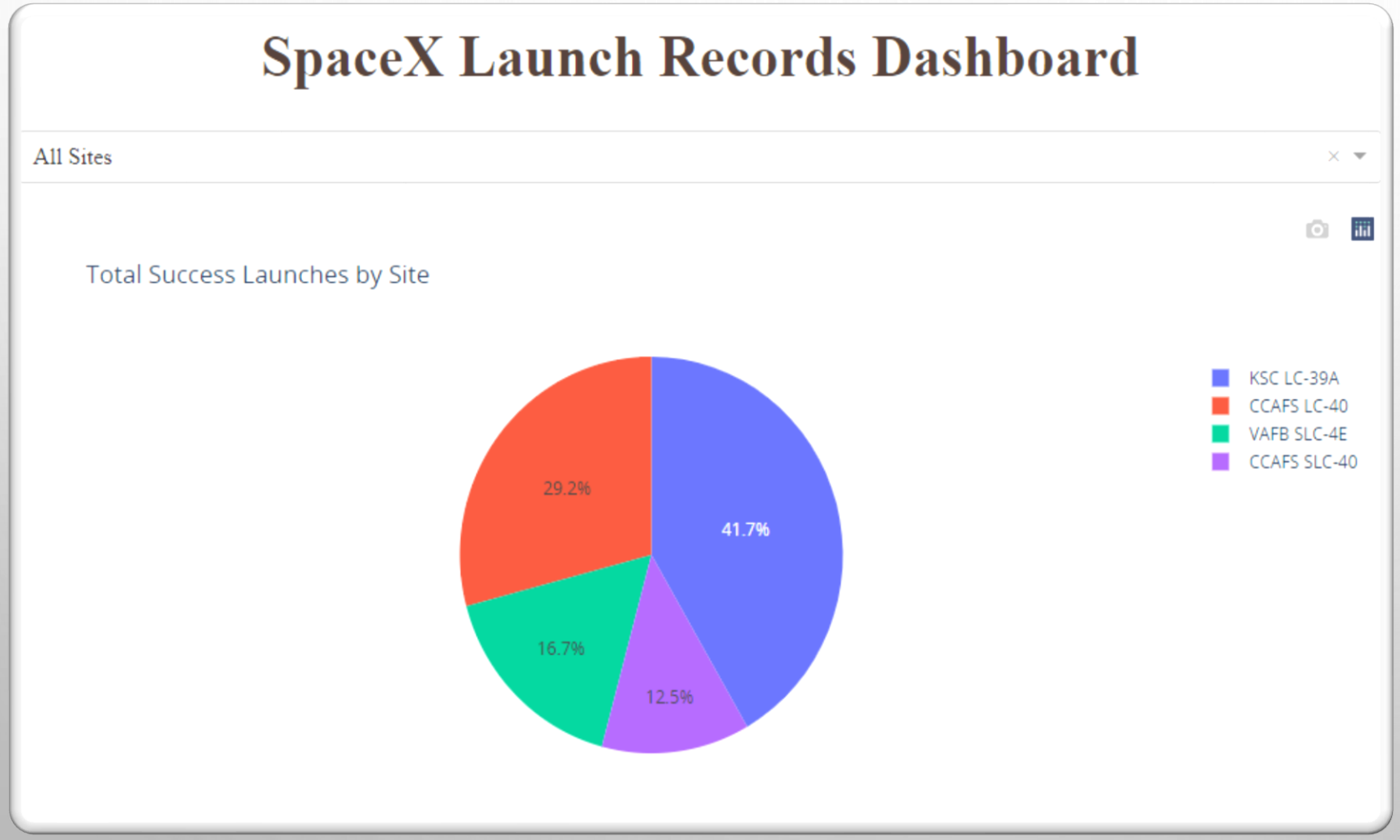




INTERACTIVE DASHBOARD WITH PLOTLY DASH



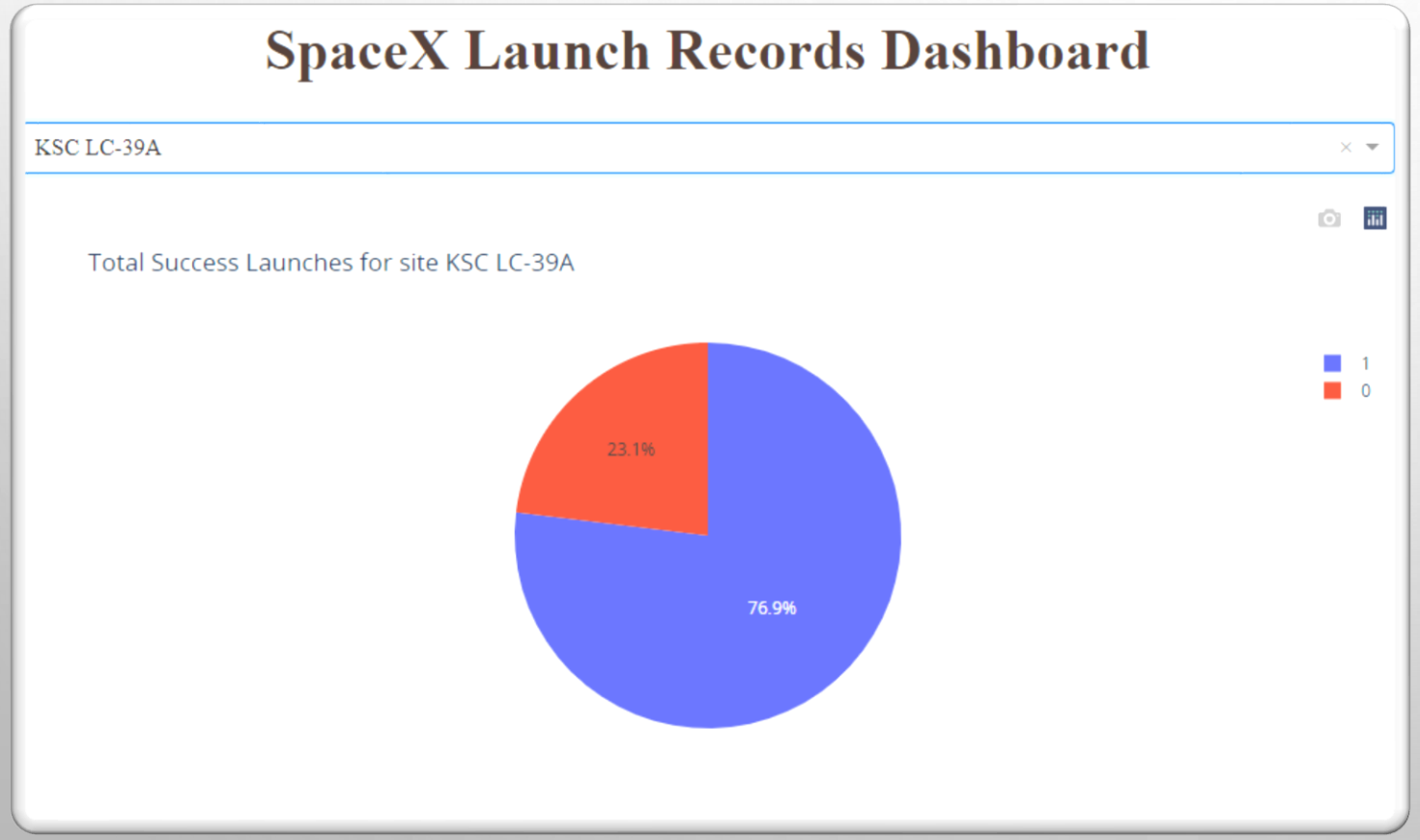
DASHBOARD TAB 1



The launch site KSC LC-39 A had the most successful launches, with 41.7% of the total successful launches.

DASHBOARD TAB 2

PIE CHART FOR THE LAUNCH SITE WITH HIGHEST LAUNCH SUCCESS RATIO



The launch site KSC LC-39 A also had the highest rate of successful launches, with a 76.9% success rate.

DASHBOARD TAB 3

- Plotting the launch outcome vs. payload for all sites shows a gap around 4000 kg, so it makes sense to split the data into 2 ranges:
  - 0 – 4000 kg (low payloads)
  - 4000 – 10000 kg (massive payloads)
- From these 2 plots, it can be shown that the success for massive payloads is lower than that for low payloads.
- It is also worth noting that some booster types (v1.0 and B5) have not been launched with massive payloads.

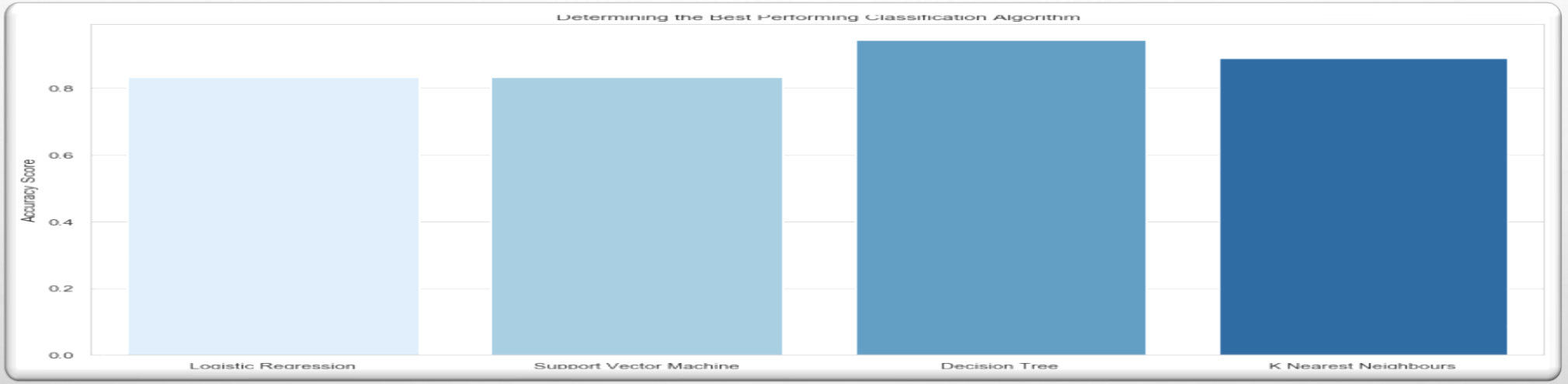
Classification Accuracy

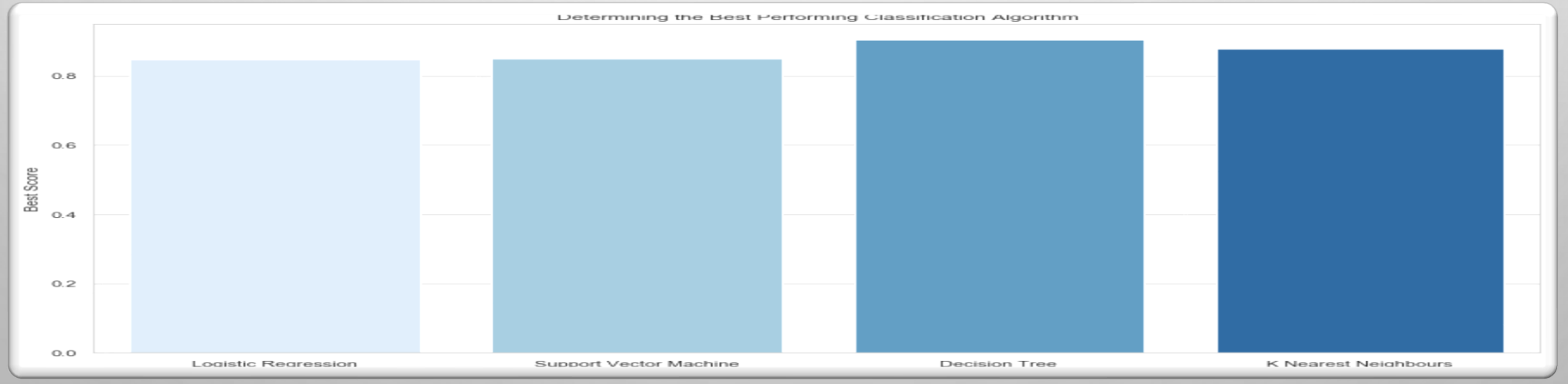
```
models = {'KNeighbors':knn_cv.best_score_,
              'DecisionTree':tree_cv.best_score_,
              'LogisticRegression':logreg_cv.best_score_,
              'SupportVector': svm_cv.best_score_}

bestalgorithm = max(models, key=models.get)
print('Best model is', bestalgorithm,'with a score of', models[bestalgorithm])
if bestalgorithm == 'DecisionTree':
    print('Best params is :', tree_cv.best_params_)
if bestalgorithm == 'KNeighbors':
    print('Best params is :', knn_cv.best_params_)
if bestalgorithm == 'LogisticRegression':
    print('Best params is :', logreg_cv.best_params_)
if bestalgorithm == 'SupportVector':
    print('Best params is :', svm_cv.best_params_)
```

```
Best model is DecisionTree with a score of 0.8732142857142856
Best params is : {'criterion': 'gini', 'max_depth': 6, 'max_features': 'auto', 'min_samples_leaf': 2, 'min_samples_split': 5, 'splitter': 'random'}
```

The decision tree classifier is the model with the highest classification accuracy

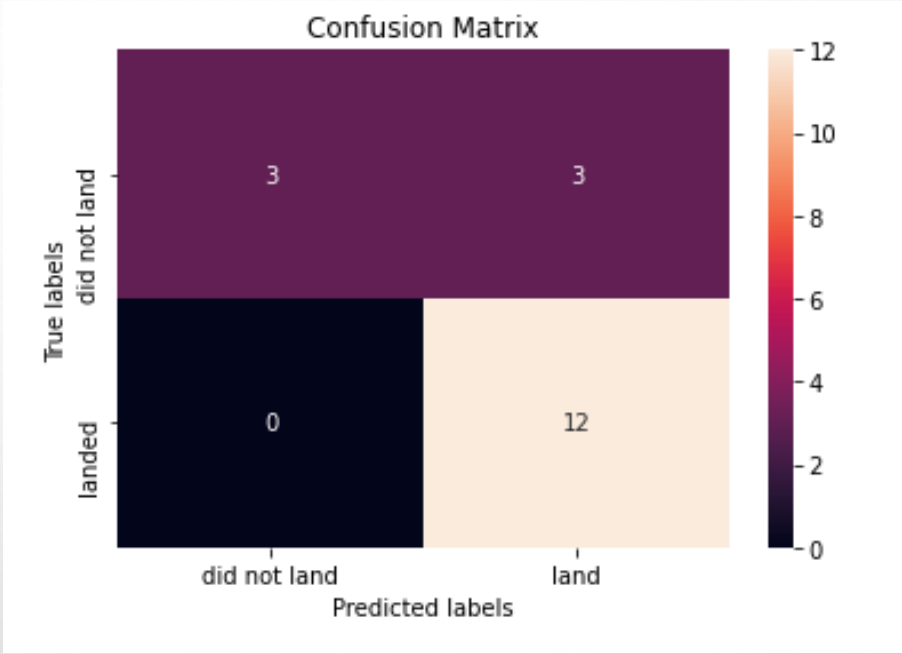
Classification Accuracy



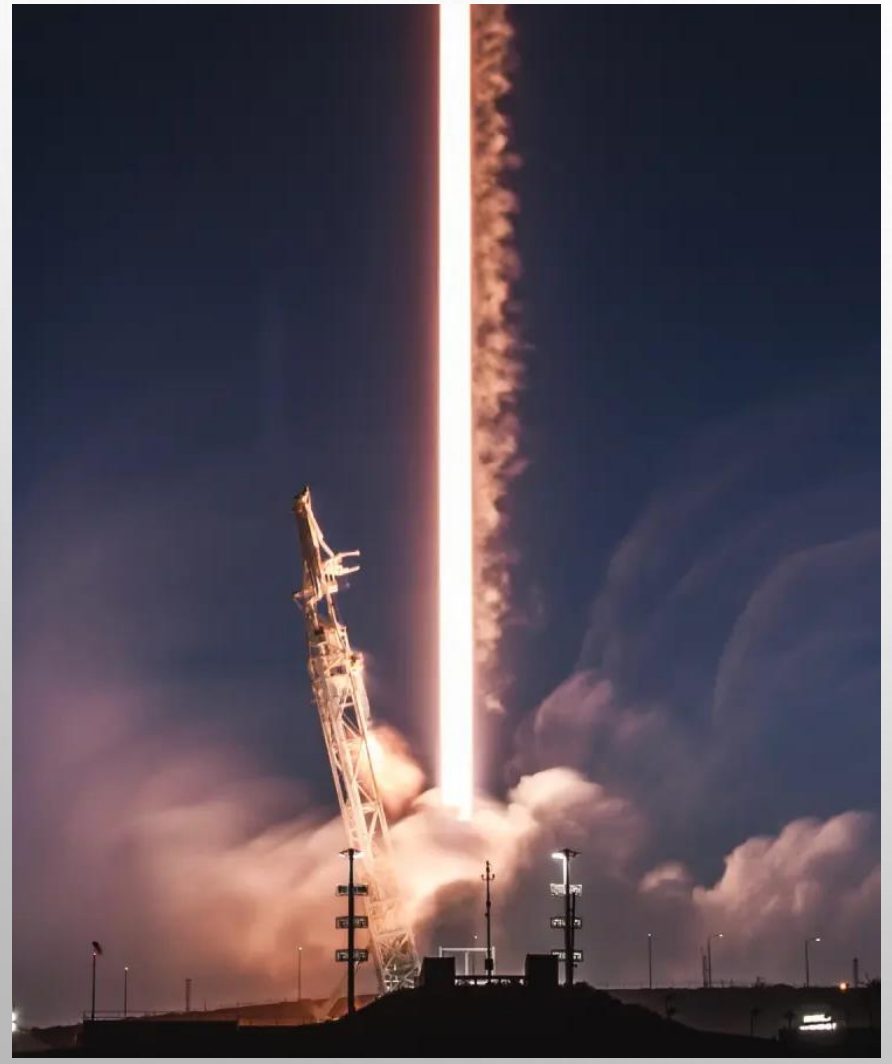


Confusion Matrix

The confusion matrix for the decision tree classifier shows that the classifier can distinguish between the different classes. The major problem is the false positives .i.e., unsuccessful landing marked as successful landing by the classifier



CONCLUSION



We can conclude that:

- The larger the flight amount at a launch site, the greater the success rate at a launch site.
- Launch success rate started to increase in 2013 till 2020.
- Orbits es-l1, geo, heo, sso, vleo had the most success rate.
- Ksc lc-39a had the most successful launches of any sites.
- The decision tree classifier is the best machine learning algorithm for this task.